

Mirostław J. Kubiak



C++

**Zadania z programowania
z przykładowymi rozwiązaniami**

WYDANIE II

Helion 

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Opieka redakcyjna: Ewelina Burska
Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/cppza2>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-3494-6

Copyright © Helion 2018

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

	Wstęp do II wydania	5
Rozdział 1.	Proste operacje wejścia – wyjścia	9
Rozdział 2.	Podajemy decyzje w programie	21
Rozdział 3.	Iteracje	33
Rozdział 4.	Tablice	63
	Tablice jednowymiarowe	63
	Tablice dwuwymiarowe	67
	Działania na macierzach	85
	Pętle zakresowe i kontenery	95
	Klasa vector	96
	Klasa deque	99
Rozdział 5.	Podprogramy	103
	Rekurencja	121
Rozdział 6.	Programowanie obiektowe	129
	Klasa osoba	142
	Hermetyzacja danych, dziedziczenie i polimorfizm	146
Rozdział 7.	Pliki tekstowe	151
	Pliki tekstowe	151

Rozdział 8.	Wskaźniki, zmienne dynamiczne i struktury danych	163
	Wskaźniki	163
	Wskaźniki i tablice	165
	Zmienne dynamiczne	170
	Zmienne dynamiczne dla tablic	171
	Struktury danych	172
Rozdział 9.	Szablony	177
	Prosty szablon dla funkcji	178
	Szablon dla różnych typów	180
	Szablony dla klas	181
	Bibliografia	184

Rozdział 1.

Proste operacje wejścia – wyjścia

W tym rozdziale zamieściłem proste zadania z przykładowymi rozwiązaniami ilustrujące, w jaki sposób komputer komunikuje się z użytkownikiem w języku C++.

Każda aplikacja powinna mieć możliwość komunikowania się z użytkownikiem. Wykorzystując proste przykłady, pokażę, jak program napisany w języku C++ komunikuje się z użytkownikiem poprzez standardowe operacje wejścia – wyjścia.

Plik nagłówkowy z instrukcji:

```
#include <iostream>
```

zawiera definicje klas¹ umożliwiających wykonywanie operacji wejścia – wyjścia na strumieniach. Do wyprowadzania danych na ekran służy standardowy strumień wyjściowy `cout`, który w języku C++ domyślnie przypisuje ekran do standardowego urządzenia wyjściowego systemu operacyjnego. Aby wyświetlić komunikat lub dane, trzeba do strumienia wyjściowego `cout` zastosować symbol podwójnego znaku mniejszości `<<` (operacja wstawiania). Dwa znaki mniejszości należy wprowadzić z klawiatury.

Do wprowadzania danych do programu służy standardowy strumień wejściowy `cin` oraz operator `>>` (dwa znaki większości, które również wprowadzamy z klawiatury), np. `cin >> a;`

¹ Więcej informacji na temat klas Czytelnik znajdzie w rozdziale 6.

W programie, dla wygody, użyłem **przestrzeni nazw** (ang. *namespace*):

```
using namespace std;
```

Dzięki temu wygodniejsze będzie posługiwanie się strumieniami wejściowymi `cin` i wyjściowymi `cout`².

Do formatowania strumienia wyjściowego będziemy używali flagi formatującej `fixed` i manipulatora `setprecision(n)`. Flaga `fixed` stosuje do liczb zmiennoprzecinkowych ustaloną kropkę dziesiętną, natomiast manipulator `setprecision(n)` ustala ich precyzję na n , np. zapis `cout << setprecision(2);` oznacza, że liczby zmiennoprzecinkowe będą wyświetlane z dokładnością do dwóch miejsc po kropce.

Zastosowanie manipulatora `setprecision(n)` wymaga włączenia do programu pliku nagłówkowego:

```
#include <iomanip>
```

Opisane powyżej podejście do operacji wejścia – wyjścia nazywa się obiektywowym³.

Zadanie

1.1

Napisz program, który oblicza pole prostokąta. Wartości boków `a` i `b` wprowadzamy z klawiatury. W programie należy przyjąć, że zmienne `a` i `b` oraz pole są typu `float` (rzeczywistego). Przyjmujemy format wyświetlania ich na ekranie z dokładnością do dwóch miejsc po kropce.

Przykładowe rozwiązanie — listing 1.1

```
// Zadanie 1.1

#include "stdafx.h"
#include <iostream>
#include <iomanip>
#include <conio.h>

using namespace std;
```

² Umieszczenie dyrektywy `using` na poziomie globalnym (na poziomie pliku) czyni zawartość przestrzeni nazw dostępną globalnie (tzn. w całym pliku) [1].

³ Więcej informacji na temat obiektowych operacji wejścia – wyjścia, flag i manipulatorów znajdzie Czytelnik w bibliografii [1, 2] lub na stronach WWW poświęconych językowi programowania C++ pod adresem <http://www.cplusplus.com/>.

```
int main()
{
    float a, b, pole;

    cout << "Program oblicza pole prostokąta." << endl;
    cout << "Podaj bok a." << endl;
    cin >> a;
    cout << "Podaj bok b." << endl;
    cin >> b;

    pole = a*b;

    fixed; //flaga
    cout << setprecision(2); //ustalenie precyzji
    cout << "Pole prostokąta o boku a = " << a << " i boku b = " << b;
    cout << " wynosi " << pole << "." << endl;

    _getch(); //naciśnij klawisz Enter

    return 0;
}
```

Linijka kodu:

```
float a, b, pole;
```

umożliwia zadeklarowanie zmiennych *a*, *b* i *pole* (wszystkie zmienne w programie są typu rzeczywistego *float*). Instrukcja:

```
cout << "Program oblicza pole prostokąta." << endl;
```

wyświetla na ekranie komputera komunikat *Program oblicza pole prostokąta*. Instrukcja *cin >> a*; czeka na wprowadzenie z klawiatury komputera liczby, która następnie zostanie przypisana zmiennej *a*. Pole prostokąta zostaje obliczone w wyrażeniu:

```
pole = a*b;
```

Za wyświetlenie wartości zmiennych *a* i *b* oraz *pole* wraz z odpowiednim opisem są odpowiedzialne następujące linijki kodu:

```
cout << "Pole prostokątna o boku a = " << a << " i boku b = " << b;
cout << " wynosi " << pole << "." << endl;
```

Flaga *fixed* używa ustalonej kropki dziesiętnej dla liczb zmiennoprzecinkowych. Zapis:

```
cout << setprecision(2);
```

oznacza, że liczby te będą wyświetlane na ekranie z dokładnością do dwóch miejsc po kropce. Natomiast funkcja:

```
_getch(); // naciśnij klawisz Enter
```

(ang. *get character* — czytaj znak) czeka na naciśnięcie klawisza *Enter* (lub na naciśnięcie dowolnego klawisza). **UWAGA!** Przed literą *g* w instrukcji `_getch()` znajduje się znak podkreślenia `_`.

Prototyp tej funkcji znajduje się w pliku nagłówkowym *conio.h*. Instrukcja:

```
endl;
```

(ang. *end of line* — koniec linii) przenosi kursor na początek następnej linii.

Komentarze w programie oznaczamy dwoma ukośnikami:

```
// to jest komentarz do programu
```

Są one ignorowane w procesie kompilacji.

Rezultat działania programu można zobaczyć na rysunku 1.1.

Rysunek 1.1.

Efekt działania programu
Zadanie 1.1

```
Program oblicza pole prostokąta.
Podaj bok a.
1
Podaj bok b.
2
Pole prostokąta o boku a = 1.00 i boku b = 2.00 wynosi 2.00.
```

Zadanie

1.2

Napisz program, który wyświetla na ekranie komputera wartość predefiniowanej stałej $\pi = 3,14\dots$. Należy przyjąć format prezentowania tej stałej, oznaczanej w języku C++ jako `M_PI`, z dokładnością do pięciu miejsc po kropce.



Wskazówka

Stałe matematyczne, w tym `M_PI`, są rozszerzeniem biblioteki standardowej `math.h`. Jeśli chcemy użyć tych stałych w programie w środowisku Microsoft Visual C++, to należy zadeklarować za pomocą preprocesora zmienną `_USE_MATH_DEFINES` przed dołączeniem pliku nagłówkowego `math.h`. Ilustrują to następujące instrukcje zamieszczone w programie:

```
#define _USE_MATH_DEFINES
#include <math.h>
```

Przykładowe rozwiązanie — listing 1.2

```
//Zadanie 1.2

#include "stdafx.h"
#include <iostream>
#include <iomanip>
#define _USE_MATH_DEFINES
#include <math.h>
#include <conio.h>

using namespace std;

int main()
{
    cout << "Program wyświetla wartość predefiniowanej stałej pi" << endl;
    cout << "z dokładnością do pięciu miejsc po kropce." << endl;
    cout << "pi = " << fixed << setprecision(5) << M_PI << endl;

    _getch(); // naciśnij klawisz Enter

    return 0;
}
```

Rezultat działania programu można zobaczyć na rysunku 1.2.

Rysunek 1.2.

*Efekt działania
programu
Zadanie 1.2*

Program wyświetla wartość predefiniowanej stałej pi
z dokładnością do pięciu miejsc po kropce.
pi = 3.14159

Zadanie**1.3**

Napisz program, który wyświetla na ekranie komputera pierwiastek kwadratowy z wartości predefiniowanej stałej $\pi = 3,14\dots$. Należy przyjąć format wyświetlania tego pierwiastka z dokładnością do dwóch miejsc po kropce.



Pierwiastek kwadratowy ze stałej π obliczamy, korzystając z funkcji `sqrt()`.
Funkcja ta znajduje się w pliku nagłówkowym `math.h`.

Przykładowe rozwiązanie — listing 1.3

```
//Zadanie 1.3

#include "stdafx.h"
#include <iostream>
#include <iomanip>
#define _USE_MATH_DEFINES
```

```

#include <math.h>
#include <conio.h>

using namespace std;

int main()
{
    cout << "Program wyświetla pierwiastek kwadratowy z pi" << endl;
    cout << "z dokładnością do dwóch miejsc po kropce." << endl;

    cout << "Sqrt(pi) = " << fixed << setprecision(2) << sqrt(M_PI) << endl;

    _getch(); // naciśnij klawisz Enter

    return 0;
}

```

Rezultat działania programu można zobaczyć na rysunku 1.3.

Rysunek 1.3.

Efekt działania programu
Zadanie 1.3

Program wyświetla pierwiastek kwadratowy z pi z dokładnością do dwóch miejsc po kropce.
Sqrt(pi) = 1.77

Zadanie

1.4

Napisz program, który oblicza objętość kuli o promieniu r . Wartość promienia wprowadzamy z klawiatury. W programie należy przyjąć, że r jest typu `double` (rzeczywistego). Dla zmiennych r oraz `objetosc` należy przyjąć format wyświetlania ich na ekranie z dokładnością do dwóch miejsc po kropce.

Przykładowe rozwiązanie — listing 1.4

```

//Zadanie 1.4

#include "stdafx.h"
#include <iostream>
#include <iomanip>
#define _USE_MATH_DEFINES
#include <math.h>
#include <conio.h>

using namespace std;

int main()
{
    double r, objetosc;

```

```

cout << "Program oblicza objętość kuli o promieniu r." << endl;
cout << "Podaj promień r." << endl;
cin >> r;
objetosc = 4*M_PI*r*r*r/3;
cout << fixed << setprecision(2);
cout << "Objętość kuli o promieniu r = " << r << " wynosi ";
cout << objetosc << "." << endl;

_getch(); // naciśnij klawisz Enter

return 0;
}

```

Objętość kuli o promieniu r oblicza następująca linijka kodu:

```
objetosc = 4*M_PI*r*r*r/3;
```

gdzie potęgowanie zostało zamienione na mnożenie.

Rezultat działania programu można zobaczyć na rysunku 1.4.

Rysunek 1.4.

Efekt działania programu
Zadanie 1.4

```

Program oblicza objętość kuli o promieniu r.
Podaj promień r.
1
Objętość kuli o promieniu r = 1.00 wynosi 4.19.

```

Zadanie

1.5

Napisz program, który oblicza wynik dzielenia całkowitego bez reszty dla dwóch liczb całkowitych: $a = 37$ i $b = 11$.



Wskazówka

W języku C++ w przypadku zastosowania operatora dzielenia $/$ dla liczb całkowitych reszta wyniku jest pomijana.

Przykładowe rozwiązanie — listing 1.5

```

//Zadanie 1.5

#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;

int main()
{
    auto a = 37, b = 11;

```

```

cout << "Program wyświetla wynik dzielenia całkowitego" << endl;
cout << "dla dwóch liczb całkowitych." << endl;
cout << "Dla liczb a = " << a << " i b = " << b << endl;
cout << a << "/" << b << " = " << a/b << "." << endl;

_getch(); // naciśnij klawisz Enter

return 0;
}

```

W standardzie C++ 11 wprowadzono udogodnienie pozwalające kompilatorowi dedukować typ zmiennej na podstawie typu wartości inicjalizującej — jest to **automatyczna deklaracja typów**. Słowo kluczowe `auto` umożliwia kompilatorowi przydzielenie zmiennej tego samego typu co wartość inicjalizująca. W ten sposób omija się deklarację typu obowiązującą w starej składni języków C i C++, np. `int a = 37`. Ilustruje to następująca linijka kodu:

```
auto a = 37, b = 11;
```

W standardzie C++ 11 również wprowadzono inną inicjalizację dla pojedynczych zmiennych. Zastosowano inicjalizator klamrowy zamiast znaku przypisania `=`. Następujące linijki kodu:

```
auto a = 37, b = 11;
```

```
i
```

```
auto a{37}, b{11};
```

są równoważne.

W tej książce stosuję wyłącznie starą składnię. Zachęcam jednak Czytelnika, aby w ramach ćwiczeń dodatkowych zastosował inicjalizator klamrowy zamiast znaku przypisania `=`.

Rezultat działania programu można zobaczyć na rysunku 1.5.

Rysunek 1.5.
Efekt działania
programu
Zadanie 1.5

```

Program wyświetla wynik dzielenia całkowitego
dla dwóch liczb całkowitych.
Dla liczb a = 37 i b = 11
37/11 = 3.

```

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

Zostań specem od C++!

C++ wciąż pozostaje jednym z podstawowych, najważniejszych i najbardziej potrzebnych języków programowania. Dobry programista powinien go znać i umiejętnie wykorzystywać do tworzenia najróżniejszych projektów. Jak najłatwiej porządnie opanować taki język? Jak przeskoczyć z poziomu podstawowego na zaawansowany? Jak nauczyć się wychwytywać niuanse decydujące o jakości proponowanych rozwiązań? Odpowiedź jest jedna: trzeba ćwiczyć, rozwiązywać kolejne zadania, mierzyć się z coraz większymi wyzwaniami, pisać kod i porównywać go z kodem pisany przez mistrzów.

Drugie, zaktualizowane wydanie tej popularnej pozycji powstało w odpowiedzi na potrzeby czytelników. Pomoże Ci doskonalić Twój warsztat w zorganizowany, przejrzysty i niebanalny sposób. Na początek wystarczy znajomość podstawowych elementów C++ oraz podstaw nawigacji w bezpłatnym środowisku Microsoft Visual Studio Community 2015 with Updates. A potem czeka Cię kilkadziesiąt zadań, po których żaden problem programistyczny nie wyda Ci się zbyt trudny.

- Proste operacje wejścia – wyjścia
- Podejmowanie decyzji w programie
- Iteracje
- Tablice
- Podprogramy
- Programowanie obiektowe
- Pliki tekstowe
- Wskaźniki, zmienne dynamiczne i struktury danych
- Szablony

C++ — praktyka czyni mistrza!

Helion

księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

Sprawdź najnowsze promocje:
🔗 <http://helion.pl/promocje>
Książki najchętniej czytane:
🔗 <http://helion.pl/bestsellery>
Zamów informacje o nowościach:
🔗 <http://helion.pl/nowosci>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-283-3494-6



9 788328 334946

Informatyka w najlepszym wydaniu

cena: 29,90 zł